

CPEN 311: Digital Systems Design

Power



Learning Objectives

This slide set is not intended to make you an expert on power-aware design.

But, you should be able to:

1. Understand the difference between static and dynamic power
2. Understand several methods for estimating power in modern FPGA CAD tools
3. Have some ideas for minimizing power if you are dissipating too much



Who Cares about Power?

Three reasons that power is important:

1. Hard to get large current into a chip (Amps of current)
 - 50W at 1.2V is 42Amps ($50/1.2 = 42$)
2. Cheaper device
 - Plastic chip package to save \$\$
 - Power must be $< 2W$
3. Portability
 - Must carry energy (Power * Time) with you
 - Energy is heavy (40 Wh / kg)
 - Low power → less energy used → lower battery weight

Google's Oregon Data Center



**100MW Power Plant
(sold by Kawasaki)**



100 Megawatts, enough to power
city of Tacoma (200,000 people)

Twin cooling plants, 4 stories

Two Ways Power is Dissipated:

Dynamic Power:

Power is dissipated every time a “node” (wire) is switched (toggled) from 0-to-1 or 1-to-0:

$$P_{\text{node}} = \frac{1}{2} C_{\text{node}} * V^2$$

Over time, add up for all nodes, at clock frequency f , with “activity” α :

$$P_{\text{dynamic}} = \alpha * f * C_{\text{total}} * V^2$$

“activity”: average number of toggles-per-node each clock cycle

Static Power:

Every transistor **leaks**, even when not doing anything

Leaking = constant current from Vdd to GND (not toggle/switching)

Historically small, but static power is now comparable to dynamic power

FPGA Power:

FPGAs are especially power-greedy:

- Programmable switches mean:
 - More transistors, more leaking
 - More transistors, more capacitance (dynamic power)
 - Longer wires, more capacitance (dynamic power)

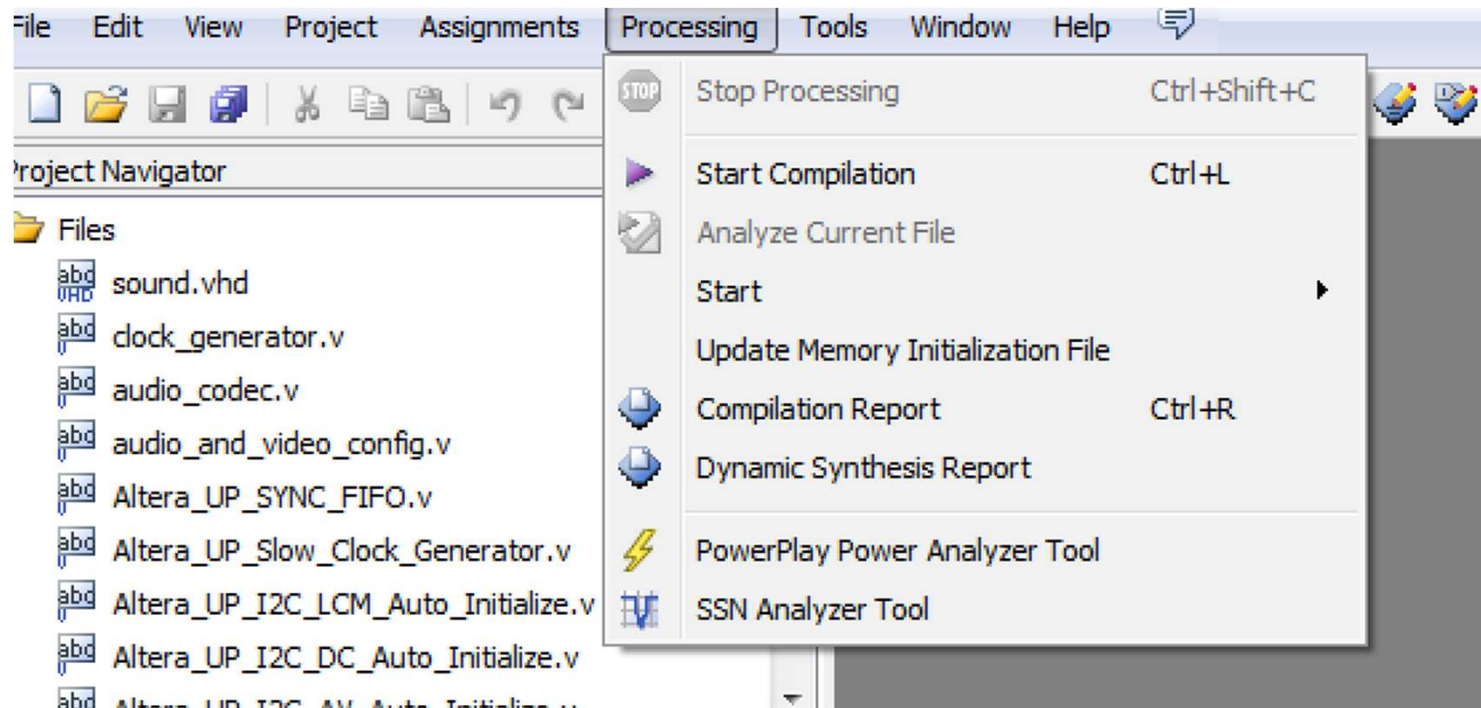
Very often, you will have a power budget:

- FPGA tools can estimate power
- Too much? need to optimize your design
(make it smaller, find algorithms with fewer wires that switch less and run in fewer clock cycles).

FPGA Power Estimation:

PowerPlay in Quartus Prime

- Provides an estimate of leakage and dynamic power



FPGA Power Estimation:

To estimate power, the tool needs to know:

1. How many transistors are on the chip (for leakage)
 - Obviously, Altera knows this so it can be hardcoded in the tool
2. How often each wire in the design is switching (α for dynamic)
 - Often called “activity”

$$P_{\text{dynamic}} = \alpha * f * C * V^2$$

The diagram illustrates the components of the dynamic power equation $P_{\text{dynamic}} = \alpha * f * C * V^2$. Arrows point from descriptive text to the variables in the equation:

- An arrow points from the text “Often called ‘activity’” to the variable α .
- An arrow points from the text “Frequency of circuit” to the variable f .
- An arrow points from the text “Capacitance of wire: proportional to length of wire” to the variable C .
- An arrow points from the text “Voltage (0.9-1.1 Volts)” to the variable V^2 .

To estimate activity α

1. Logic simulation – need accurate testbenches
2. Statistical estimation – uses probabilities, less accurate

FPGA Power Estimation:

The screenshot shows the 'Input file' section of the FPGA Power Estimation software. A red oval highlights the checkbox 'Use input file(s) to initialize toggle rates and static probabilities during power analysis' and the 'Add Power Input File(s)...' button. To the right of this section, red text states: 'If this is *not* checked, use statistical estimation method'. Below this, the 'Output file' section has a checkbox 'Write out signal activities used during power analysis' and an 'Output file name:' field. The 'Default toggle rates for unspecified signals' section includes a 'Default toggle rate used for input I/O signals:' field set to '12.5'. The 'Default toggle rate used for remaining signals' section has two radio button options: 'Use default value: 12.5' and 'Use vectorless estimation'. Red arrows point from the text 'Very Simple (fixed activity for each wire)' to the 'Use default value: 12.5' option, and from 'Statistical Method' to the 'Use vectorless estimation' option. At the bottom, there is a 'Cooling Solution and Temperature...' field, a green progress bar, a timer showing '00:00:04', and 'Start' and 'Stop' buttons.

Input file

☐ Use input file(s) to initialize toggle rates and static probabilities during power analysis

Add Power Input File(s)...

Output file

☐ Write out signal activities used during power analysis

Output file name:

Default toggle rates for unspecified signals

Default toggle rate used for input I/O signals: 12.5

Default toggle rate used for remaining signals

☐ Use default value: 12.5

☒ Use vectorless estimation

Cooling Solution and Temperature...

00:00:04

Start Stop

If this is *not* checked, use statistical estimation method

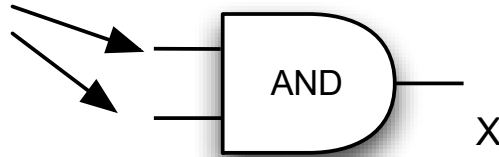
Very Simple (fixed activity for each wire)

Statistical Method

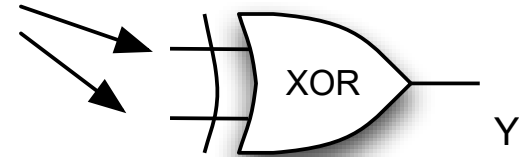
What is the statistical method?

Thought experiment: Over a long period, which signal X or Y has a higher activity (switches more often):

Random uncorrelated
input patterns



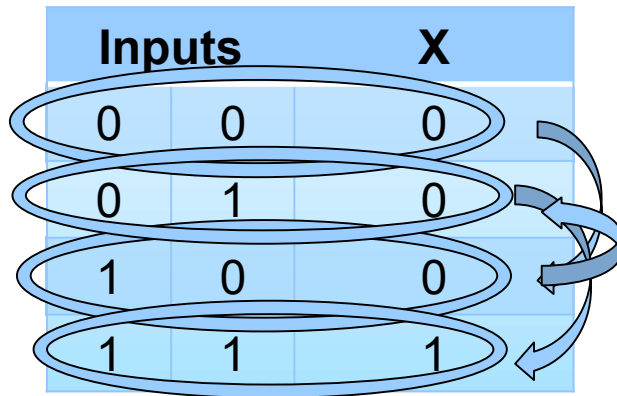
Random uncorrelated |
input patterns



Inputs.		X
0	0	0
0	1	0
1	0	0
1	1	1

Inputs		X
0	0	0
0	1	1
1	0	1
1	1	0

Inputs		X
0	0	0
0	1	0
1	0	0
1	1	1



During operation, inputs will randomly change. We can think of this as switching randomly from one line in the truth table to another

Chances that a random swap changes the output from a 0 to 1:

$$\text{Prob}(\text{the output was 0}) * \text{Prob}(\text{the output becomes 1}) = 0.75 * 0.25 = 0.1875$$

Chances that a random swap changes the output from a 1 to 0:

$$\text{Prob}(\text{the output was 1}) * \text{Prob}(\text{the output becomes 0}) = 0.25 * 0.75 = 0.1875$$

Chances that the output changes each cycle = $0.1875 + 0.1875 = 0.375$

Inputs		X
0	0	0
0	1	1
1	0	1
1	1	0

During operation, inputs will randomly change. We can think of this as switching randomly from one line in the truth table to another

Chances that a random swap changes the output from a 0 to 1:

$$\text{Prob}(\text{the output was 0}) * \text{Prob}(\text{the output becomes 1}) = 0.5 * 0.5 = 0.25$$

Chances that a random swap changes the output from a 1 to 0:

$$\text{Prob}(\text{the output was 1}) * \text{Prob}(\text{the output becomes 0}) = 0.5 * 0.5 = 0.25$$

Chances that the output changes each cycle = $0.25 + 0.25 = 0.5$

So, we expect XOR toggles more often

We can do this for all gates

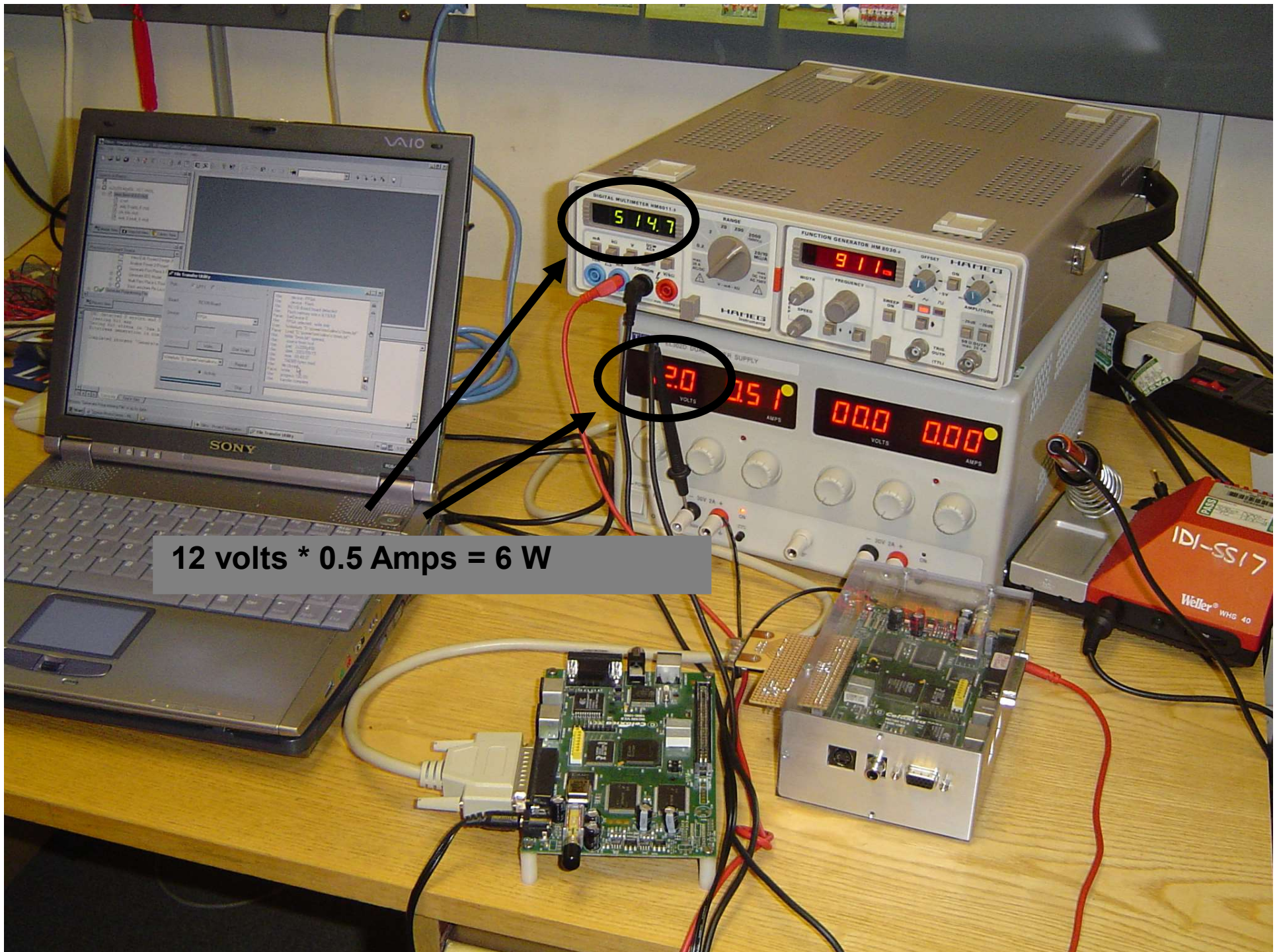
- figure out the probability that each node toggles each cycle
 - The probability of being on each line of a truth table is not always equal (in our example, we assumed 25% chance of being in each line of the truth table)
 - Depends on the gates that feed the gate we are analyzing
 - An iterative method can calculate activities similar to what we have done here
 - But the intuition is the same

This is **statistical analysis**, and is an alternative to finding the activities via **simulation**.

FPGA Power Estimation:

Table of Contents		PowerPlay Power Analyzer Summary	
Flow Log		PowerPlay Power Analyzer Status	Successful - Sat Mar 29 14:44:11 2014
Analysis & Synthesis		Quartus II 32-bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Fitter		Revision Name	sound
Assembler		Top-level Entity Name	sound
TimeQuest Timing Analyz		Family	Cyclone II
PowerPlay Power Analyz		Device	EP2C35F672C6
Summary		Power Models	Final
Settings		Total Thermal Power Dissipation	135.86 mW
Indeterminate Toggle		Core Dynamic Thermal Power Dissipation	18.67 mW
Operating Conditions		Core Static Thermal Power Dissipation	80.01 mW
Thermal Power Dissip		I/O Thermal Power Dissipation	37.18 mW
		Power Estimation Confidence	Low: user provided insufficient toggle rate data

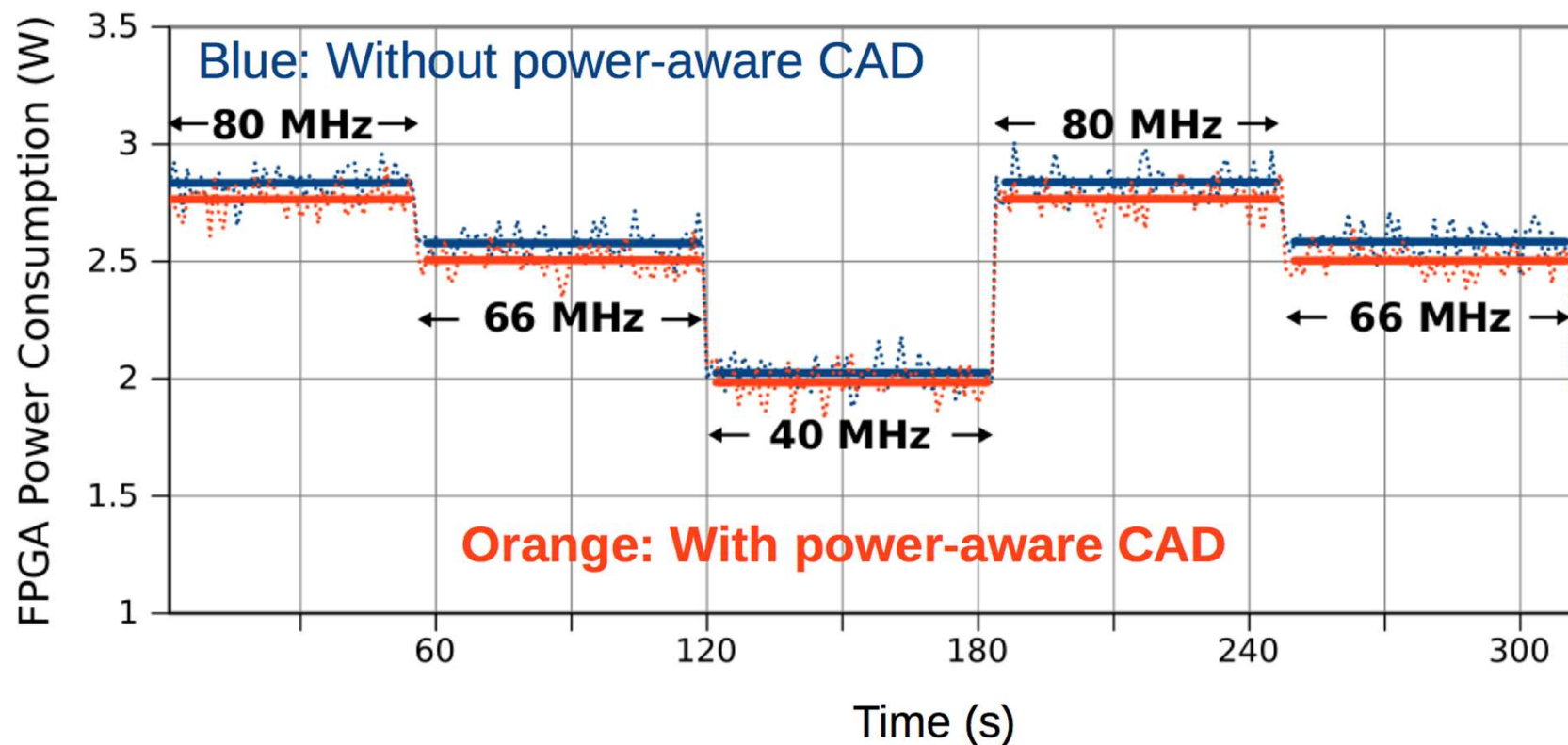
Warning: Power estimation from these tools is not very accurate, especially if you use statistical methods to estimate activity



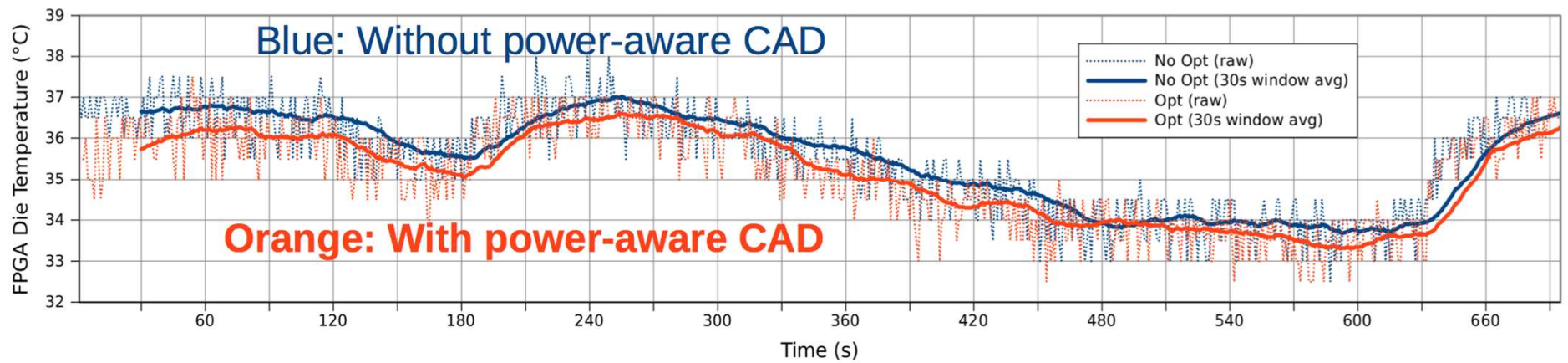
12 volts * 0.5 Amps = 6 W

Power Measurement:

Modern FPGAs have embedded power measurement



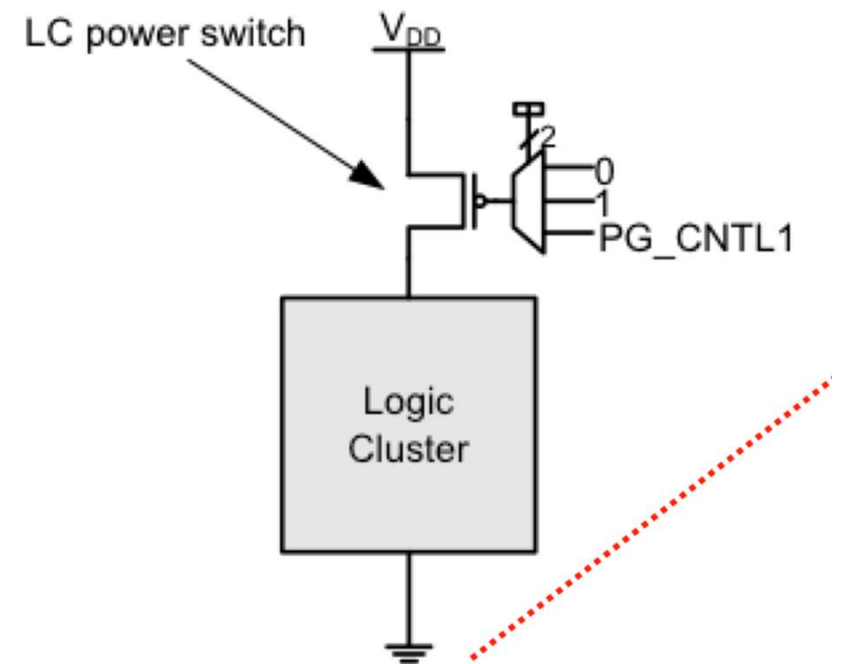
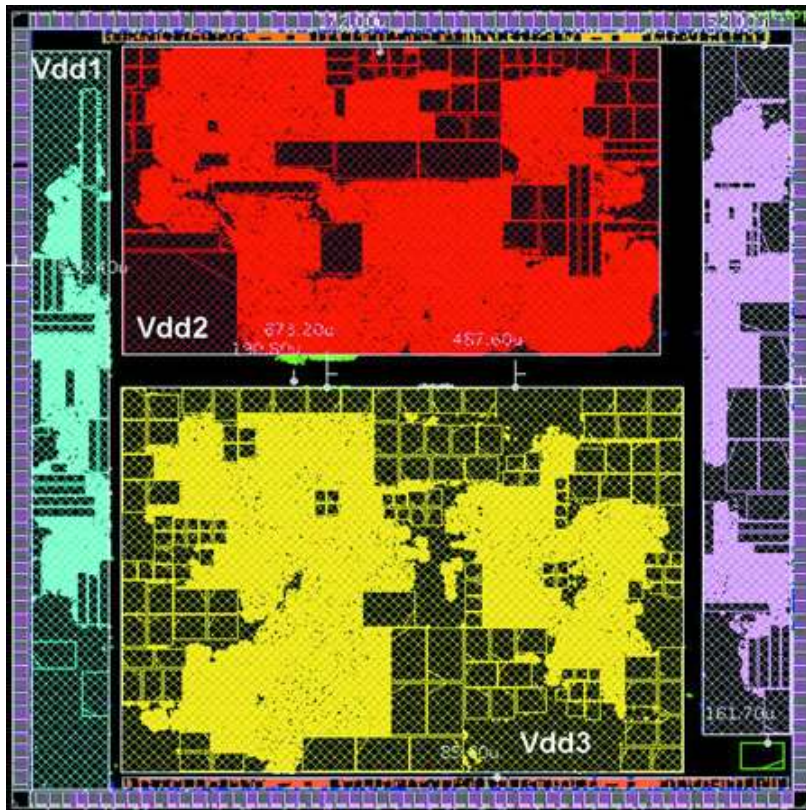
They can be used to measure temperature too:



What to do about Power?

If you are designing a custom chip, there are lots of options:

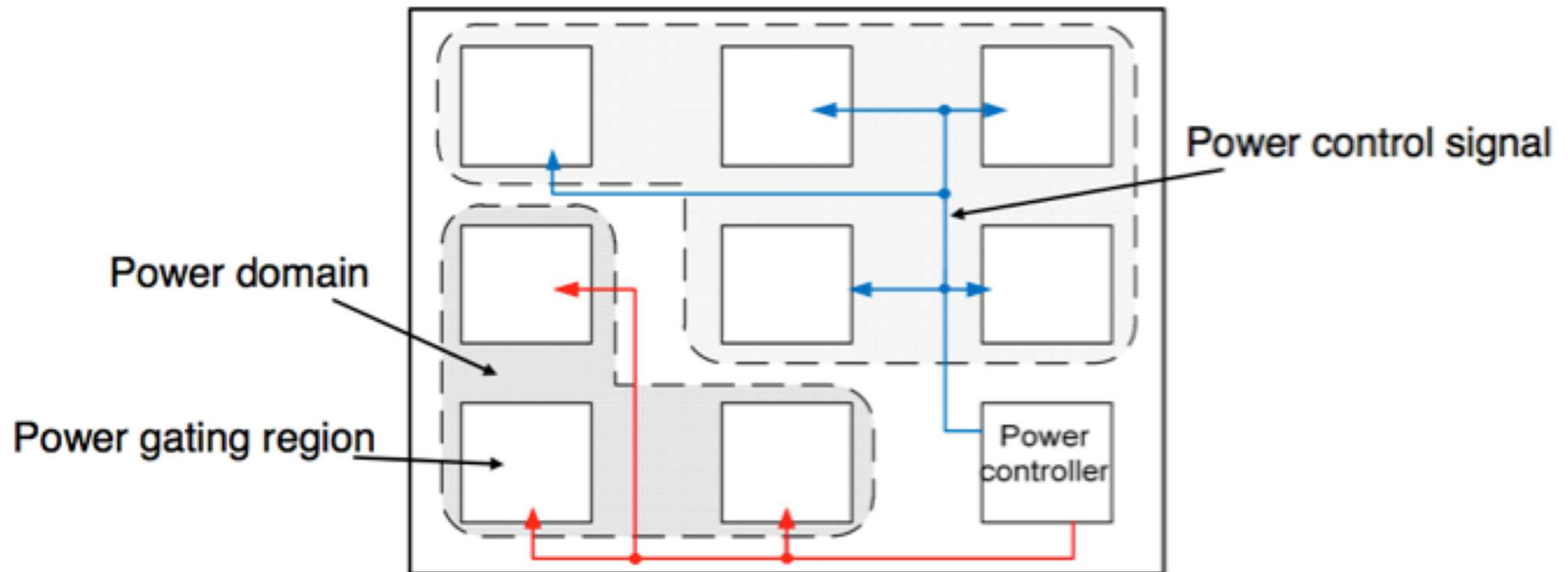
- Lower voltage, Voltage Islands
- “Dark Silicon” (turn off regions when they are not used)
- Transistors with different “threshold” voltages → slower, leak less



Can we do this in an FPGA? Today, FPGAs do not offer this feature.

However, most FPGAs allow the hard blocks to be powered down when they are not being used (eg. memories, DSP blocks, etc).

Dynamic power gating is an active research area:



(this is from a research paper)

Clock Gating

Clock distribution networks consume a lot of power

- Clock signals drive a lot of flip-flops
- The wires are long
- The clock drivers (PLLs) use a significant amount of power

All FPGAs provide “clock gating” which allows the tool to turn off the clock network to parts of the FPGA that are unused.

Xilinx: this saves 10-15% of power on typical designs

Compared to power gating:

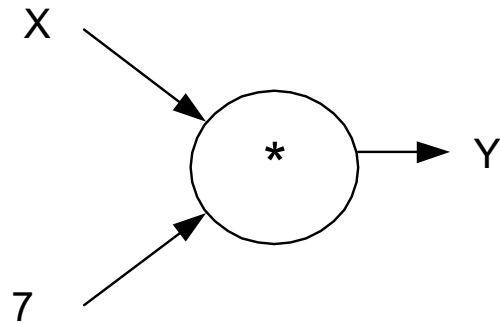
- Reduces dynamic power consumption, not static
- Clock gating is less effective than power gating could be

What to do about Power?

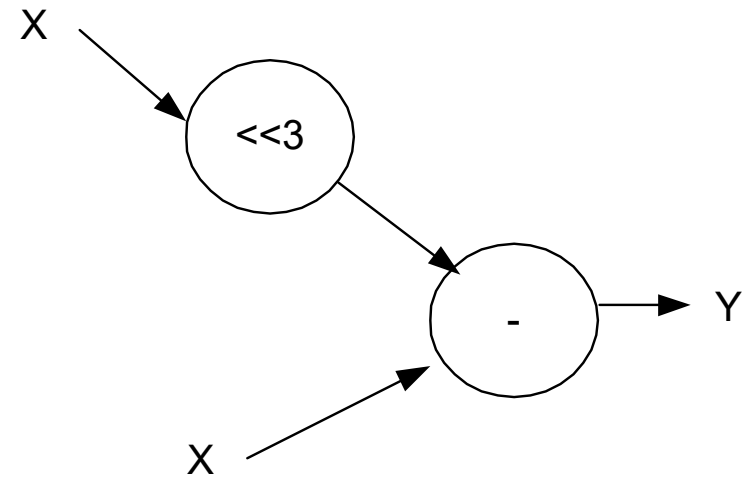
If you are using an FPGA, there are options:

- Lower voltage (some devices offer voltage range 0.9v – 1.1v)
- Try to minimize area
 - Less leakage, may allow for a smaller FPGA
- Think about restructuring your algorithm...

Simple example: Suppose we want to multiply by 7:



$$Y = 7X$$

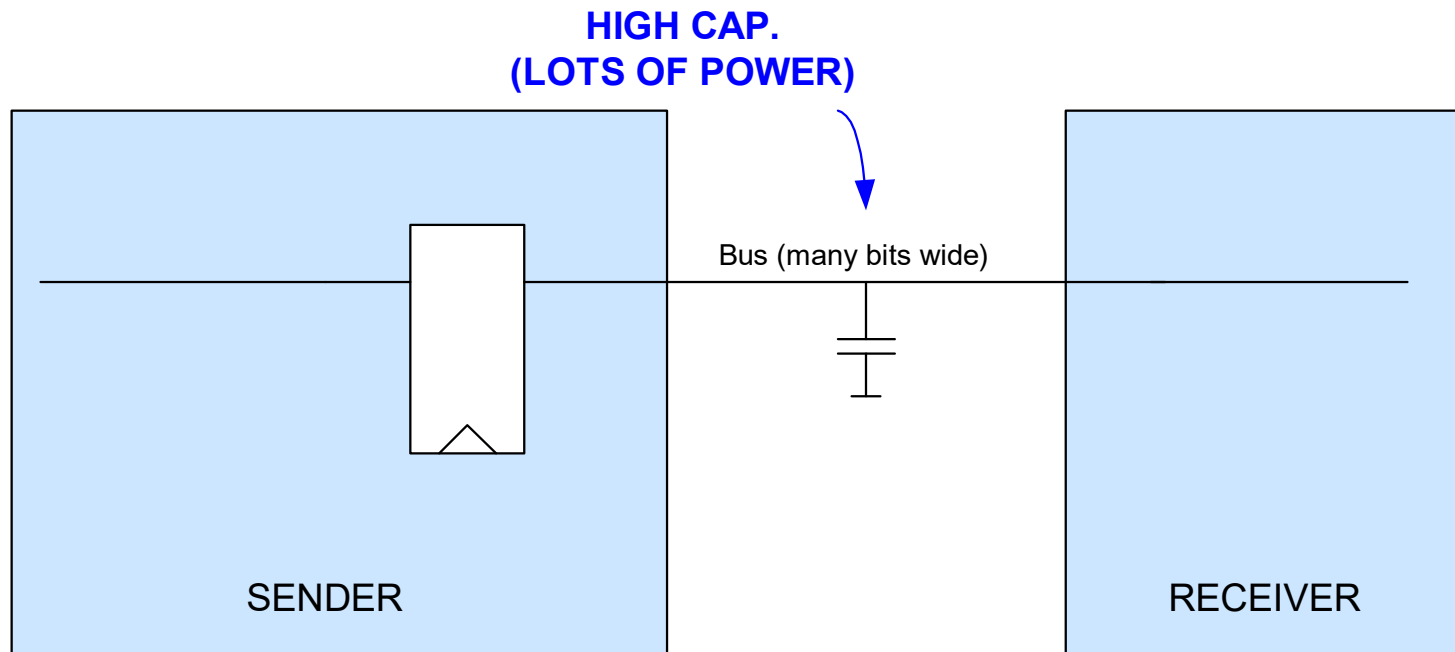


$$Y = (X \ll 3) - X$$

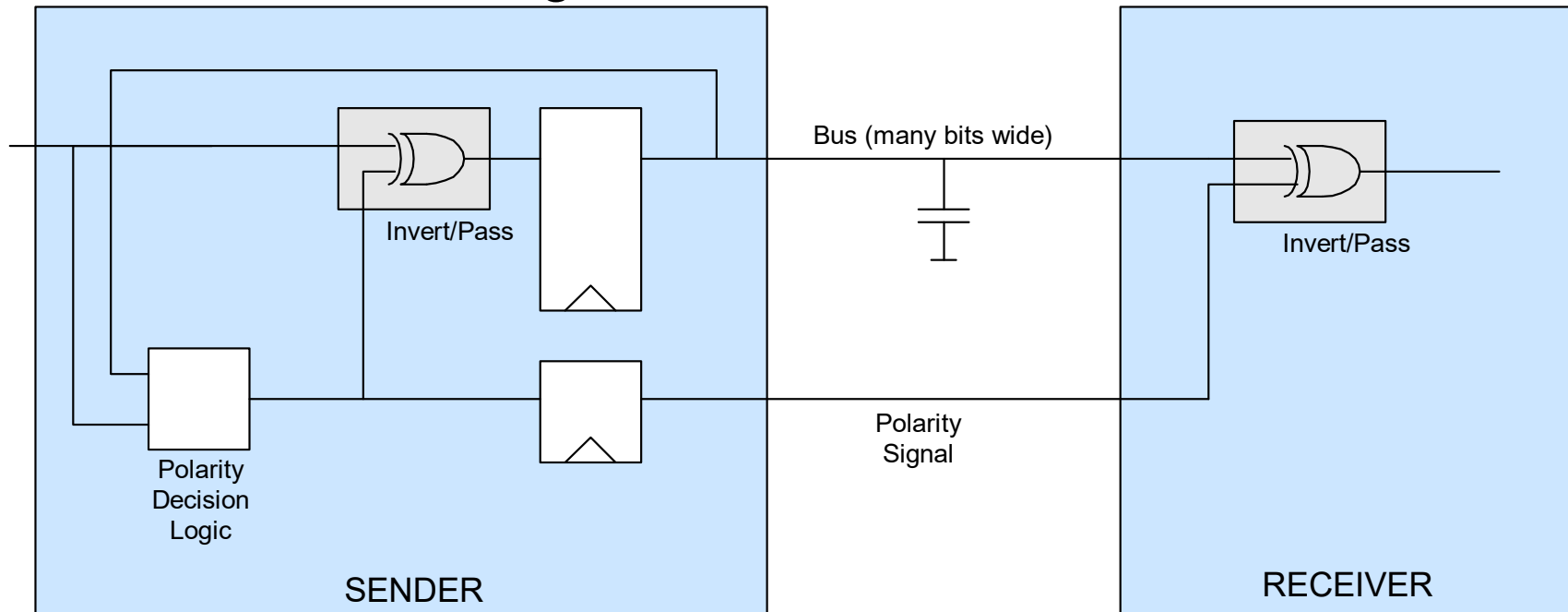
We could replace it with a left-shift by 3 and a subtraction

Another example:

Consider how you send data over a long distance:



Bus Invert Encoding:



If it would use less power to send inverse of bus, do so.

Example: if we just sent 0000, and we are now to send 0111, send 1000 instead

There is no recipe for these sorts of things, but you can think about them when you design your datapath.

It is up to your abilities as a designer

- Experience experience experience...

Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



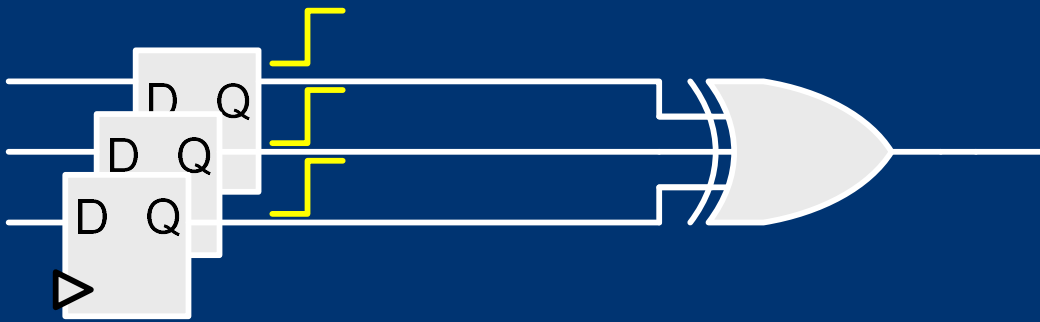
Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



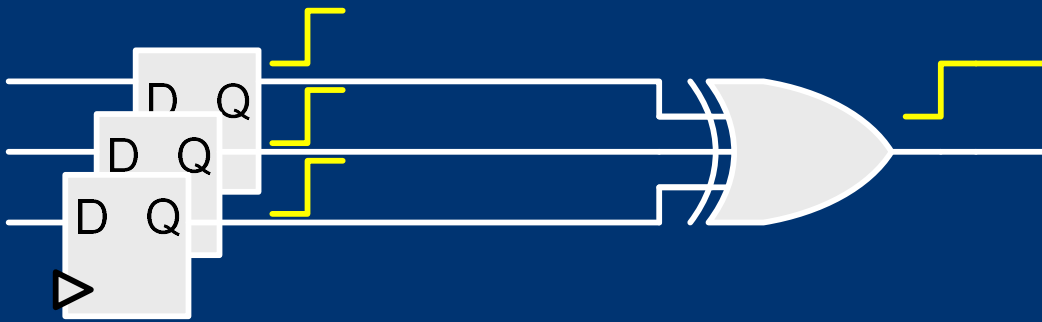
Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



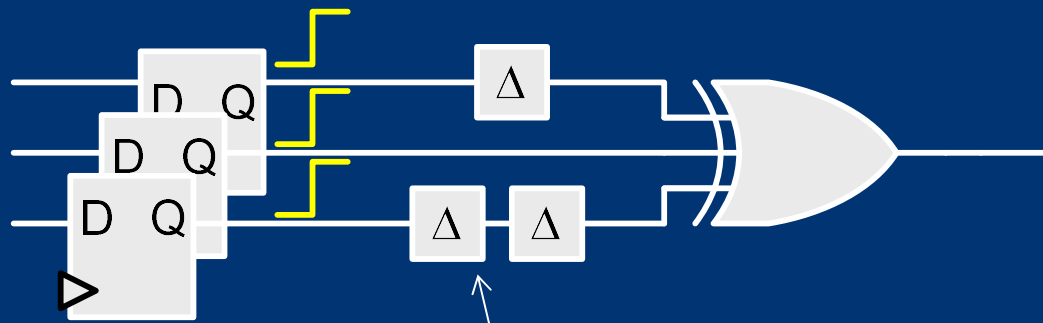
Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



Pipelining and Energy:

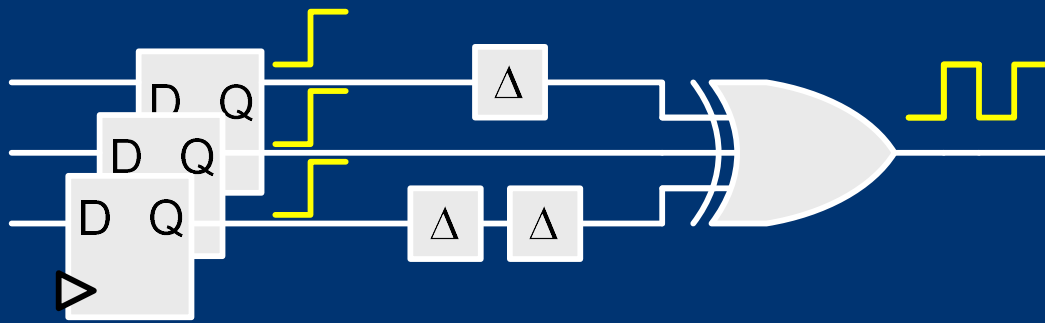
Intuitively, pipelining should reduce glitch power:



These are meant to represent delays in the wire. Due to unequal routing delays.

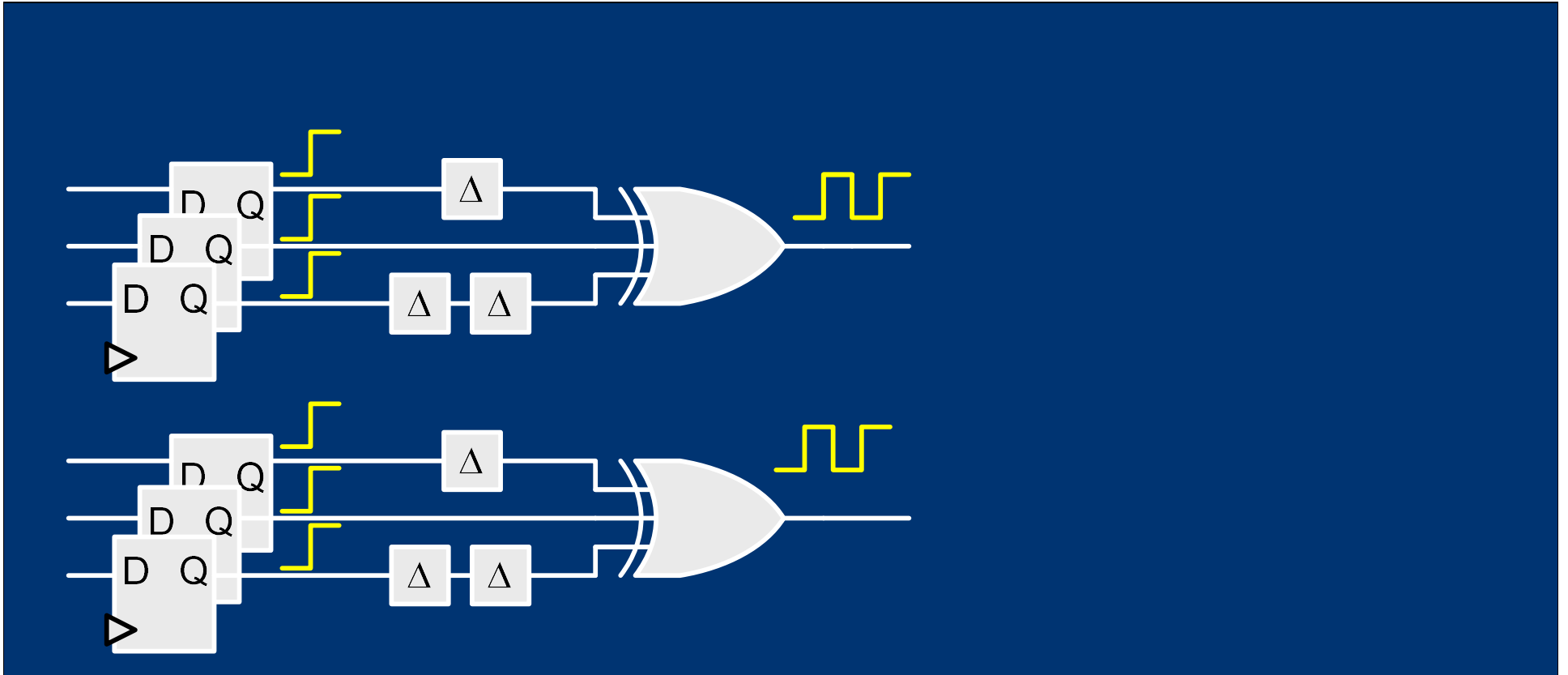
Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



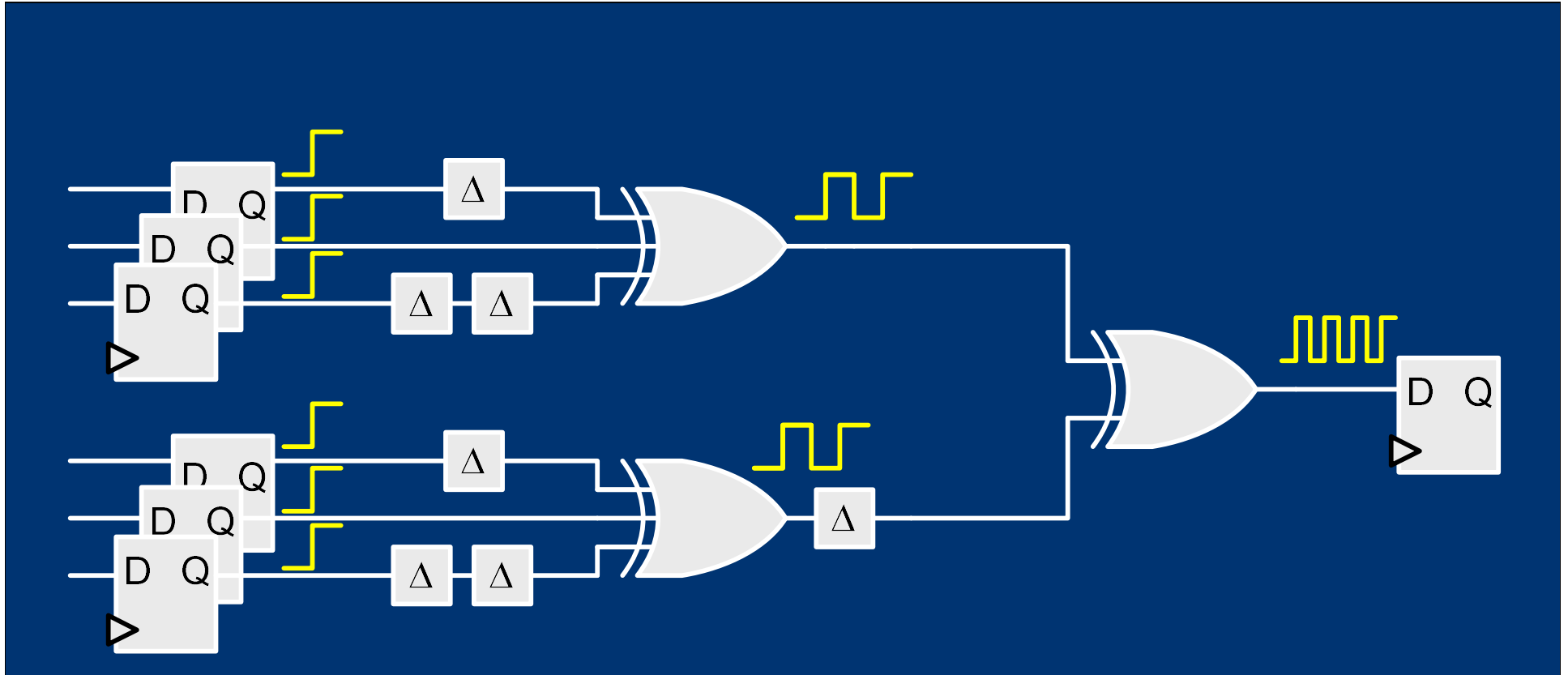
Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



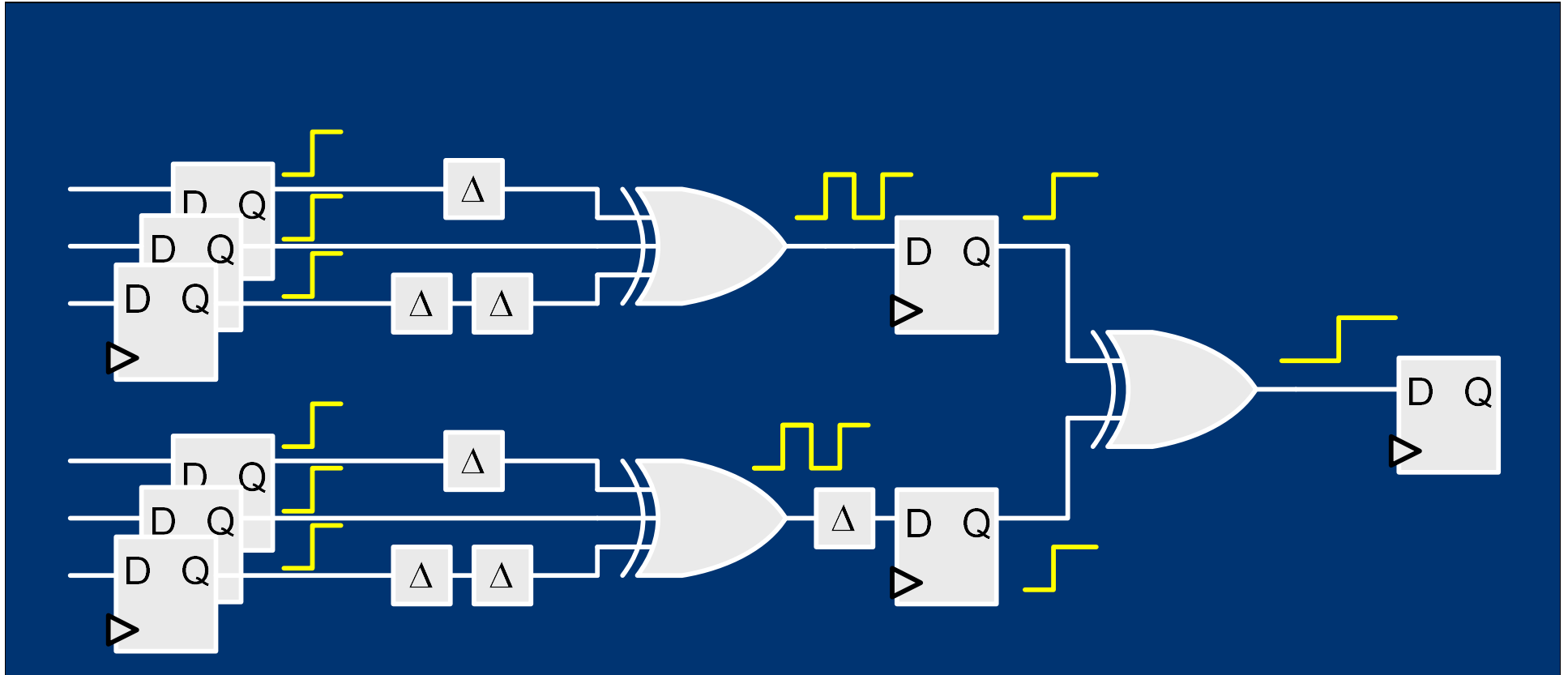
Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



Pipelining and Energy:

Intuitively, pipelining should reduce glitch power:



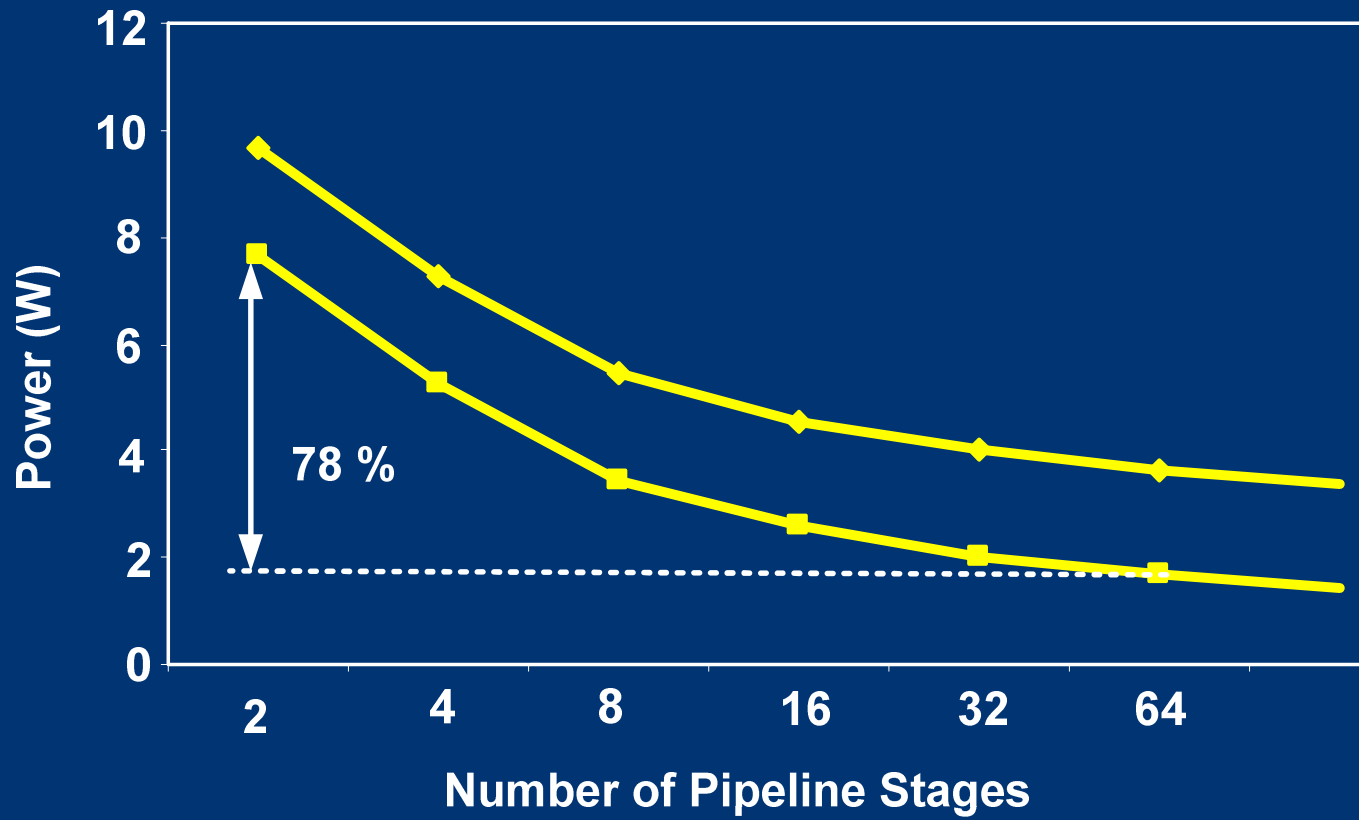
In this case, I put 2 flip-flops in to pipeline the design.
It reduced glitches on the output of the flip-flop

Intuitively, adding flip-flops prevents glitches from propagating, thereby reducing power.

Remember: Quartus Prime won't pipeline for you, but you can do it in your Verilog.

Have you make sure your design is tolerant of the extra latency, because pipelining does change behaviour

64-Bit Multiplier:



But, too much pipelining could hurt:

1. Extra flip-flops consume power as they switch
2. Extra burden on the clock tree

Why is this particularly interesting for an FPGA?

1. Wire delays can be long and switch slowly
 - leads to lots of glitches
2. Flip-flops are almost “free”, since they are there in the logic blocks anyway
3. Pipeline stages in the routing fabric

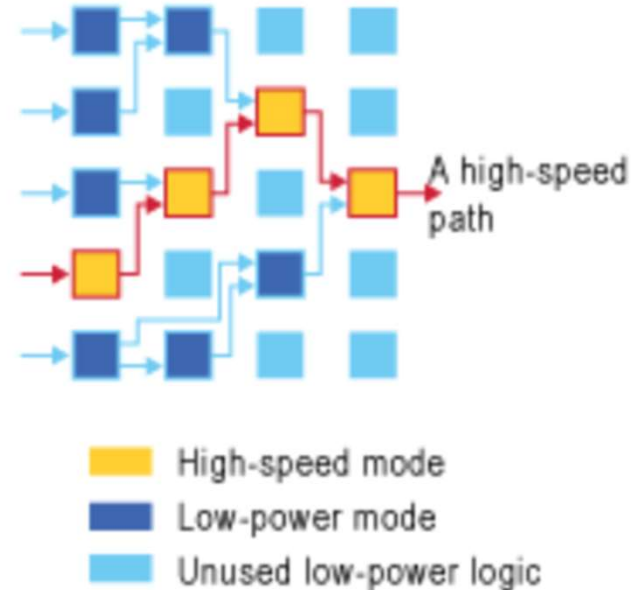
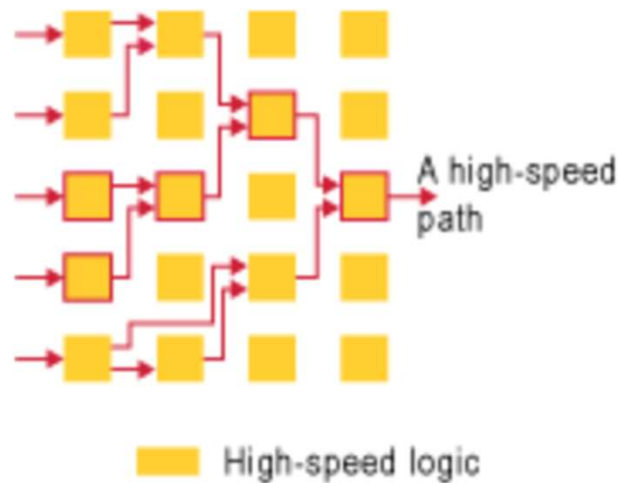
Another way to get rid of glitches...

Use glitch-free logic

What to do about Power?

CAD Tools will help:

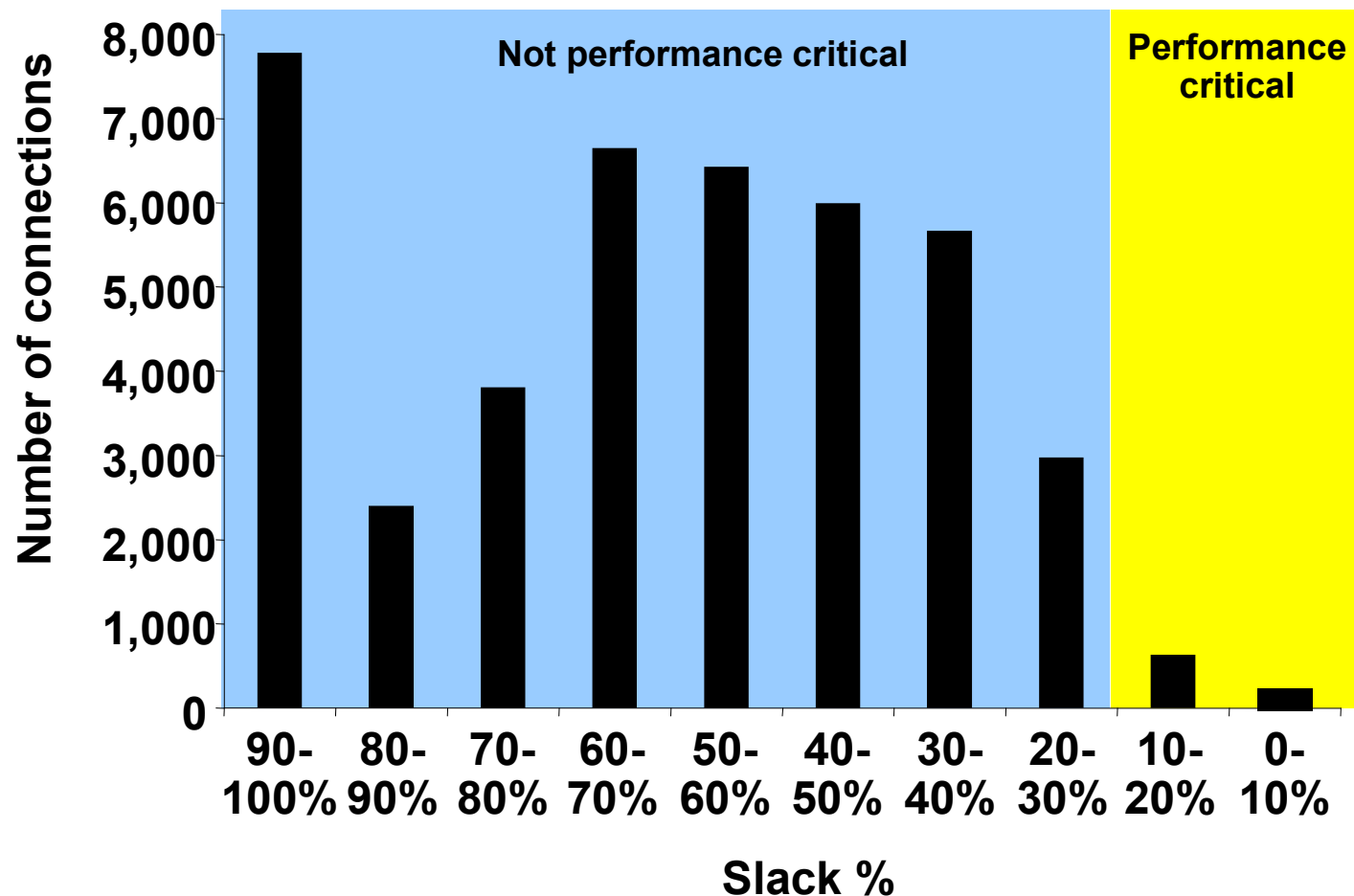
- Tries to **keep high activity nets short**
 - Affects both placement and routing
- Mapping to LUTs: try to **restructure to minimize activity**
- Take advantage of architecture features:



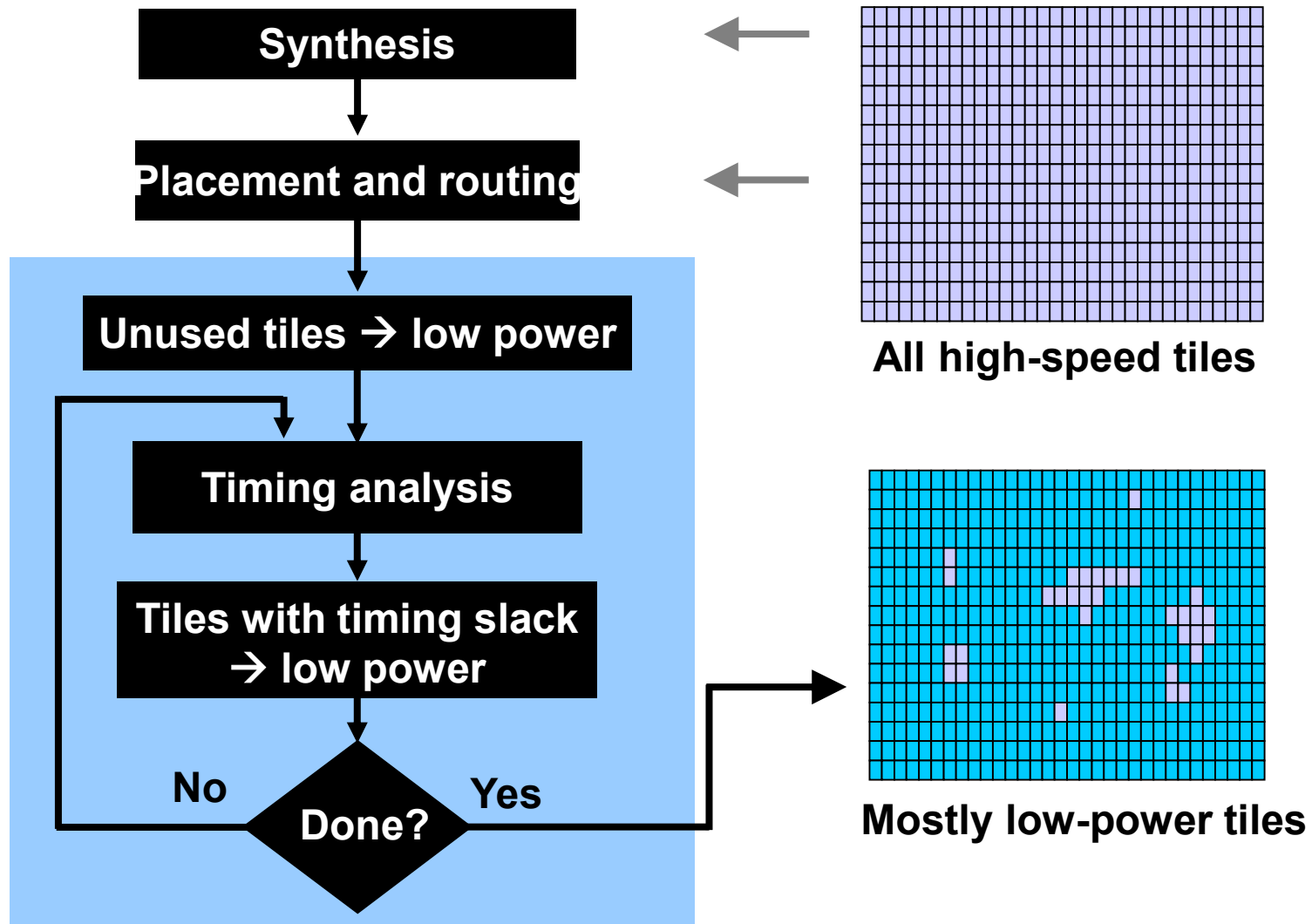
Software-Controlled Back Bias

Only a small portion of the design is speed-critical

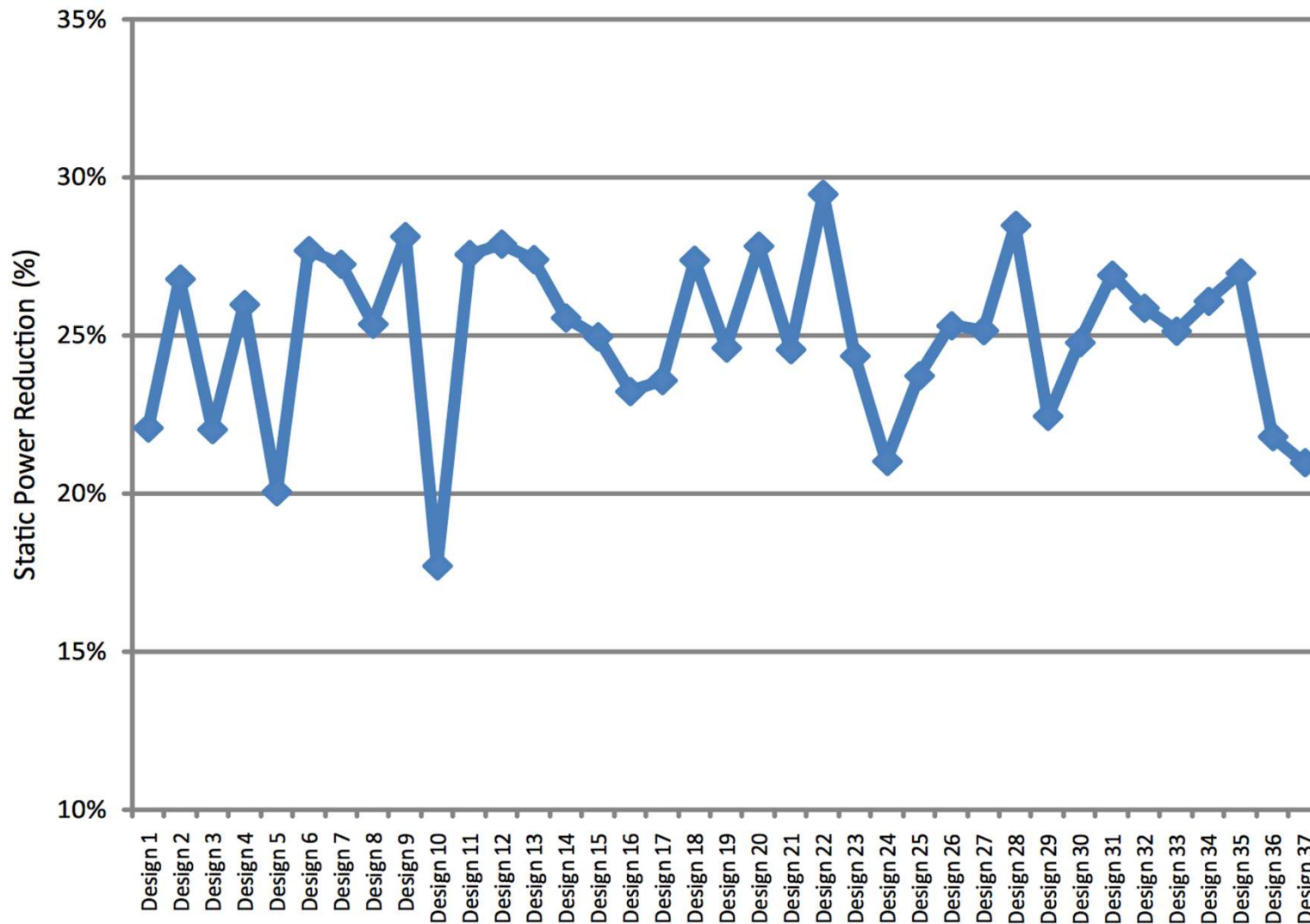
Slack Histogram



Software-Controlled Back Bias (Altera)



How much does it help? This slide is from Altera



Good news: Quartus II takes care of this for you!

Learning Objectives

This slide set is not intended to make you an expert on power-aware design.

But, you should be able to:

1. Understand the difference between static and dynamic power
2. Understand several methods for estimating power in modern FPGA CAD tools
3. Have some ideas for minimizing power if you are dissipating too much